Developer-Centric Knowledge Mining from Large Open-Source Software Repositories (CROSSMINER)

Alessandra Bagnato[3], Konstantinos Barmpis[5], Nik Bessis[7], Luis Adrián Cabrera Diego[7], Juri Di Rocco, Davide Di Ruscio[1], Tamás Gergely[4], Scott Hansen[11], Dimitris Kolovos[5], Philippe Krief[8], Ioannis Korkontzelos[7], Stéphane Laurière[9], Jose Manrique Lopez de la Fuente[10], Pedro Maló[6], Richard F. Paige[5], Diomidis Spinellis[2], Cedric Thomas[9], Jurgen Vinju[12]

[1] University of L'Aquila, Italy
{juri.dirocco.davide.diruscio}@univaq.it
[2] Athens University of Economics and Business, Greece
dds@aueb.gr
[3] Softeam R&D Department, France
alessandra.bagnato@softeam.fr
[4] FrontEndART Ltd., Hungary
tamas.gergely@frontendart.com
[5] Department of Computer Science, University of York, UK
{konstantinos.barmpis,dimitris.kolovos,richard.paige}@york.ac.uk
[6] Unparallel Innovation, Lda, Portugal
pedro.malo@unparallel.pt
[7] Department of Computer Science, Edge Hill University, UK
{nik.bessis,diegol,yannis.korkontzelos}@edgehill.ac.uk
[8] Eclipse Foundation
philippe.krief@eclipse.org
[9] OW2 consortium
{cedric.thomas,stephane.lauriere}@ow2.org
[10] Bitergia
jsmanrique@bitergia.com
[11] The Open Group
s.hansen@opengroup.org
[12] Centrum Wiskunde and Informatica
Jurgen.Vinju@cwi.nl

Abstract. Deciding if an OSS project meets the required standards for adoption is hard, and keeping up-to-date with a rapidly evolving project is even harder. Making decisions about quality and adoption involves analysing code, documentation, online discussions, and issue trackers. There is too much information to process manually and it is common that uninformed decisions have to be made with detrimental effects. CROSSMINER aims to remedy this by automatically extracting the required knowledge and injecting it into the developers' Integrated Development Environments (IDE), at the time they need it to make design decisions. This allows them to reduce their effort in knowledge acquisition and to increase the quality of their code. CROSSMINER uniquely combines advanced software project analyses with online IDE monitoring. Developers will be monitored to infer which information is timely, based on readily available knowledge stored earlier by a set of advanced offline deep analyses of related OSS projects.

1 Project data

- **Acronym:** CROSSMINER[13]
- **Title:** Developer-Centric Knowledge Mining from Large Open-Source Software Repositories
- **Partners:** The Open Group — *Project Coordinator*, University of York, University of L'Aquila — *Technical Coordinator*, Edge Hill University, Centrum Wiskunde & Informatica, Athens University of Economics and Business, UNPARALLEL, Softeam, Frontendart, Bitergia, OW2 consortium, Eclipse Foundation Europe GmbH
- **Start date:** 1 January 2017, **Duration:** 36 months

2 Introduction

Open-source software (OSS) is computer software distributed with a license that allows access to its source code, free redistribution, the creation of derived works, and unrestricted use [6]. Unlike commercial software which is typically developed within the context of a particular organisation with a well-established business plan and commitment to the maintenance, documentation and support of the software, OSS is very often developed in a public, collaborative, and loosely-coordinated manner. This has several implications to the level of quality of OSS software as well as to the level of support that OSS communities provide to users of the software they produce. Consequently, *developing new software systems by reusing existing open source components* raises challenges related to at least the following activities [14]: *i)* searching for candidate components, *ii)* evaluating a set of retrieved candidate components to find the most suitable one, and *iii)* adapting the selected components to fit the specific requirements.

Dependence on OSS projects can either be a blessing or a curse. The ability to accurately assess the risks and benefits of adopting particular OSS projects as components is essential to the software development community at large. The EU OSSMETER FP7[5] project developed a distributed and horizontally-scalable platform for incremental analysis of multiple dimensions of open-source software projects including their source code, communication channels, and bug tracking systems. The aim of CROSSMINER is to extend the outcomes of the OSSMETER project and to deliver an integrated open-source platform that will support the *development of complex software systems* by (1) enabling *monitoring, in-depth analysis and evidence-based selection* of open source components, and (2) facilitating *knowledge extraction* from large open-source software repositories.

The paper is structured as follows: Section 3 gives an overview of the CROSSMINER project. Section 4 outlines the planned evaluation process and concludes the paper.

3 The CROSSMINER approach

Figure 1 shows a high-level overview of the CROSSMINER approach. It shows two major use cases and two minor user channels which are implemented using two archi-

[13] `http://www.crossminer.org`

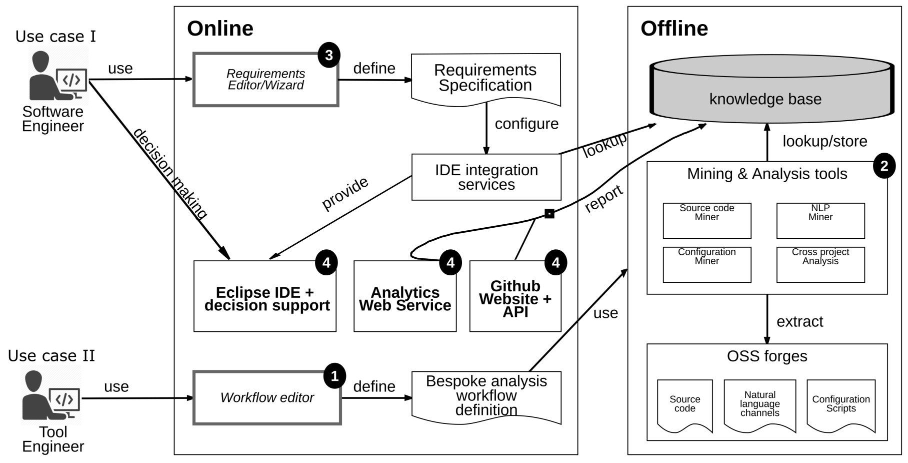

Fig. 1. CROSSMINER approach at a glance

tectural stages: online and offline. We describe the two major use cases here in some detail to clarify what CROSSMINER entails as a whole.

▷ In step ❶ the tool engineers of Use case II use a domain-specific (graphical) editor in their IDE to compose new workflows of data sources and computations. This functionality is commonly available in big data analytics suites; here we specialise this functionality for typical OSS project analysis tasks.

▷ Mining and analysis tools will run incrementally in step ❷, and possibly on a remote server, to extract relevant information from a pre-configured set of projects and a list of projects configured by the software engineers of Use case I.

▷ The software engineers of Use case I have a wizard to configure CROSSMINER with a rich set of requirements (step ❸), which includes not only registering a set of projects of interest but also expressing preferences regarding the algorithms and processes used to project the mined information into the IDE. This configuration is an important step to make meaningful assessment possible later, since it makes the context and preferences of the engineer explicit to the platform in terms of technological, quality, configuration, and licensing aspects.

▷ Finally, step ❹ is when the acquired information is put to action, actively supporting the engineers via the IDE, managers via the web site, and the open-source community via GitHub integration.

Typical examples of IDE services which may be introduced or enhanced using this architecture are:

▷ *code assist* — propose relevant code snippets, ranked by relevance and quality and informed by the earlier configuration;

▷ *infer/Fix project setup* — retrieve a list of ranked relevant reusable components, then set up relevant projects in the IDE and configure dependent projects to use them;

▷ *monitoring of development activities* of the engineers, who will be notified of relevant facts pertaining to their current task context.

In the following the scientific and technological objectives to be achieved for realizing the approach shown in Fig. 1 are summarized.

3.1 Development of source code analysis tools

State of the art: Source code analysis has its firm fundaments in compiler (front-end) construction [3] and reverse engineering [23]. Based on this theory and technology, to extract meaningful and accurate metrics, we developed reusable front-ends generating informative reusable intermediate models with the OSSMETER project (for the Java and PHP languages). Recently, examples of source code analyses have been scaled up to acquire information over large source code and software install bases [9,16]. However, mining information from source code at scale, made available in an integrated platform, including the acquisition, extraction, and querying of source code from groups of arbitrarily selected projects is just beyond the current state-of-the-art. Especially when the platform should cater for bespoke analyses based on the intermediate models there exists few related work in this regard [9].
Innovation: Mining source code artefacts to actively support decision making by software engineers *inside* their IDE requires scaling the technology for source code analysis to a level where we can mine in much larger corpora on the one hand, and on the other hand can enable much more context-specific (bespoke) analyses. At the same time the non-functional requirement of scalability must not imply a lower expected level of accuracy of the (bespoke) analyses. To scope this challenge in balancing trade-offs, and making it manageable for the current project we reason back from example decisions and the information required to make them. The main focus will be on *dependency management*: to help software engineers which (parts of) open-source components to depend on and how to manage these dependencies.

3.2 Development of natural language analysis tools

State of the art: Text mining tools to automatically extract, analyse, summarise and assess information found in communication channels and bug trackers related to OSS are valuable for supporting OSS development. Although there is a significant amount of literature analysing code repositories and communication channels, there are only very few attempts to use these sources to help programmers as they program or to improve their output. For example, similarity methods have been proposed to identify the most relevant Stack Overflow discussions to the code that a developer is working on in an IDE and recommend them to improve developing performance [19,20]. Microsoft has just released Bing Developer Assistant for Visual Studio[14], which searches GitHub repositories, locates and presents examples of API usage relevant to the code being developed in Visual Studio.
Innovation: In CROSSMINER we plan to provide software developers with text analysis components integrating three innovative aspects: *i)* a *user-oriented platform*, allowing users to tailor analysis to their needs by synthesising components into workflows. We will design and implement text mining components to identify the types of bugs

[14] `http://visualstudiogallery.msdn.microsoft.com/a1166718-a2d9-4a48-a5fd-504ff4ad1b65`

and discussions in communication channels associated with an OSS project. Developers will be able to select the components of interest and synthesise them into workflows. Depending on the selected components the output will contain different information useful for the developers; *ii)* we will investigate methods for using word embeddings for *representing text* in the domain of discussions about OSS; *iii) new sources*, such as social media and Stack Overflow, and the analysis of *code snippets*.

3.3 Development of system configuration analysis tools

State of the art: The practices, principles, and tools associated with Infrastructure as Code (IaC) and the analysis of software configuration management systems are in nascent phase. Studies to explore the characteristics of configuration code written in languages such as Puppet and Chef are scarce. Similarly, tools to carry out analyses of system configuration code have just started to emerge. Jiang et al. [13] study the co-evolution of Puppet and Chef configuration files with source, test, and build code. They analyse the software repositories of 256 OpenStack projects and distinguish files as infrastructure, which contain configuration code in Puppet or Chef language, production, build, and test. They find that configuration code comes in large files, changes more frequently, and presents tight coupling with test files. Sharma et al. [22] carry out an empirical study of 4,621 Puppet repositories to understand the characteristics of configuration code written in Puppet. Puppet Forge[15] is the repository of Puppet modules and provides an evaluation of configuration code quality through a quality score based on three aspects: code quality score provided by Puppet-Lint[16], compatibility with Puppet, and metadata quality. On the other hand, although empirical studies have examined the build aspect of software configuration management [2,1,17,4,18,10], the corresponding results have not yet been adopted by software developers.
Innovation: CROSSMINER aims to significantly improve the state of the art in the configuration management domain by introducing advanced analysis techniques to process configuration code and other relevant artefacts. In particular, *collecting meta-data and computing various metrics* is the first step towards a comprehensive analysis. CROSSMINER aims to analyse configuration code written in various system configuration management languages including Puppet, Chef, and CFEngine as well as software configuration management metadata. Such source-code analysis will provide a uniform and comprehensive set of metrics that could be used to reveal the characteristics of configuration management systems. *Combining metrics and metadata* collected from configuration code with the results of source-code and natural language analysis using advanced static analysis techniques will fetch interesting insights and actionable results. *Interactive visualisation techniques* will be employed to engage the users in an effective and productive manner. Thus, suitable dashboard with DevOps-level information will be developed to show relevant metrics and insights about the analysed systems.

3.4 Development of workflow-based knowledge extractors

State of the art: OSS forges such as GitHub, GitLab and SourceForge and bug tracking tools such as Bugzilla and JIRA provide REST APIs with which users can perform

[15] `http://forge.puppetlabs.com` [16] `http://puppet-lint.com`

queries (as well as some updates) on remote data (e.g. repository metadata, bug reports). To protect the underlying systems from uncontrolled data harvesting, many of these REST APIs impose key-based rate limits that clients cannot exceed and attempts to work around them (e.g. using multiple accounts/keys) can result to network-level blocking of the offending network endpoints. In addition to making use of remote APIs to extract knowledge from open-source projects, the wide adoption of distributed version control systems (predominately Git) where the entire history of repositories can be easily cloned, has triggered the appearance of a number of tools (e.g. Gitana [7], Gitstats[17]) that can analyse locally-cloned repositories and extract and present general-purpose metrics such as development activity over time / contributions per developer etc. While such metrics are useful, more advanced knowledge extraction (e.g. such as the one conducted in [15] which measures the adoption of different model-based technologies in Github-based open-source projects) typically requires bespoke analysis which includes the use of remote APIs, cloning and local analysis of repositories, natural language processing, HTML scraping, regular expressions etc.
Innovation: In CROSSMINER we envision the development of a framework that can support the development of declarative and efficient OSS project analysis workflows. Using the envisioned framework, engineers will be able to plug together OSS data harvesting, analysis and transformation components and define their dependencies and interactions at a high level of abstraction. The framework will provide built-in support for recurring concerns such as network/API error recovery and data caching so that engineers can focus on the core analysis of the workflows, thus enhancing both productivity and maintainability. The framework will ship with robust built-in components for extracting information from widely-used systems such as Git(Hub), GHTorrent [11], Bugzilla, JIRA, NNTP and StackOverflow and will also provide extensibility mechanisms through which engineers can integrate additional components. We will also develop a set of hybrid textual/graphical editors and viewers through which engineers will be able to define knowledge extraction workflows, and also debug and monitor their execution at a high level of abstraction.

3.5 Development of cross-project relationship analysis tools

State of the art: Over the last decade several platforms have been introduced to support automated analysis of open source software. All of them provide techniques and tools to analyse projects individually and do not mine projects relationships that instead can give more insight about existing OSS components. The most representative analysis platforms are OSSMETER, Qualipso[18], Qualoss[19], Flossmetrics [20], SQO-OSS (Alitheia Core)[21], Openhub[22], and RISCOSS [23]. Also, many OSS forges (e.g., SourceForge and GitHub) provide built-in measurement facilities for the OSS projects they host.
Innovation: In CROSSMINER we envision the development of advanced techniques able to investigate relationships among different open source projects and properly organise them in a dedicated knowledge base. Beyond the typical project dependency and

[17] http://gitstats.sourceforge.net [18] http://cordis.europa.eu/project/rcn/80465_en.html
[19] http://cordis.europa.eu/project/rcn/79759_en.html [20] http://dl.acm.org/citation.cfm?id=1545011.1545457
[21] http://cordis.europa.eu/project/rcn/79362_en.html [22] http://www.openhub.net
[23] http://www.riscoss.eu

conflict relationships we aim at identifying and managing additional ones e.g., license compatibility, API compatibility, etc. A general way to represent project relationships will be devised in order to enable relevant features including the following: *i)* support for automated classification of OSS projects and discovery of related projects based on source code, configuration code, licensing, communication channel and bug tracking system analysis; *ii)* adoption of clustering mechanisms supporting multidimensional classification of OSS projects; *iii)* support for issuing notifications when quality indicators of selected OSS projects fall below a user-defined level; *iv)* support for suggesting OSS projects that can be alternatively used instead of OSS components, which have been previously selected and integrated in the software being developed.

3.6 Development of advanced integrated development environments

State of the art: Most of the current IDEs include a wide range of features to enhance developer productivity from various code completion and refactoring actions to style and error corrections. For Eclipse, the most notable related plug-ins are Codetrails Connect Community Edition[24] and Eclipse Code Recommenders[25]. These plugins learn how to use a new API from the source code of other applications or from watching how experienced developers use it, and share this information among team members through functions like code completion or snippet search. There are more novel approaches to extend the abilities of modern IDEs, to enhance programmer productivity and coding quality. For instance, in [21] authors proposed a novel approach that, given a context in the IDE, automatically retrieves pertinent discussions from StackOverflow, and evaluates their relevance. Another example is the Adinda approach (developed by van Deursen et al. [8]) that re-thinks IDE features as web services to facilitate informal inter-project communication and collaboration. Hora and Valente [12] developed a tool that helps API comparison based on compatibility and popularity information of GitHub projects. The concept of the Change-Oriented Programming Environment (COPE) research project[26] is to monitor software changes in real-time and provide actionable feedback to the developer through the IDE.
Innovation: As the above examples show, there are many different ways to give real-time suggestions to developers within their accustomed IDE. CROSSMINER brings a whole new dimension to the advanced IDEs because it collects, processes and stores a huge amount of data about open source components in a complex and cross-project data model. This enables intelligent recommendations to be provided to the developer, by going far beyond the current "code completion-oriented" practice. Our Eclipse plug-in for CROSSMINER will be developed primarily with the objective in mind that it improves the productivity of developers *in real-time* and *transparently*. Furthermore, CROSSMINER will learn from past recommendations and feedback from the developer so that even more relevant help will be given after being in use for a certain time.

[24] http://marketplace.eclipse.org/content/codetrails-connect-community-edition

[25] http://marketplace.eclipse.org/content/eclipse-code-recommenders

[26] http://cope.eecs.oregonstate.edu/

4 Evaluation and Conclusions

In this paper we provided an outline of CROSSMINER's envisioned technical contributions. The techniques and tools will be assessed by considering the needs of six end-user partners (in the domains of IoT, multi-sector IT services, API co-evolution, software analytics, software quality assurance, and OSS forges). The full chain of retrieval, analysis and presentation of results will be implemented on large-scale open-source forges like Eclipse and OW2 to assist users and demonstrate the benefits of the solution. The technical outcomes of the project as well as the evaluation results will be the subject of follow-up publications.

Acknowledgement

The research described has been carried out as part of the CROSSMINER Project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 732223.

References

1. Adams, B., De Schutter, K., Tromp, H., De Meuter, W.: The evolution of the Linux build system. Electronic Communications of the EASST 8 (2008)
2. Adams, B., Tromp, H., De Schutter, K., De Meuter, W.: Design recovery and maintenance of build systems. In: Software Maintenance, 2007. ICSM 2007. IEEE International Conference on. pp. 114–123. IEEE (2007)
3. Aho, A.V., Sethi, R., Ullman, J.D.: Compilers: Principles, Techniques and Tools. Addison-Wesley (1988)
4. Al-Kofahi, J.M., Nguyen, H.V., Nguyen, A.T., Nguyen, T.T., Nguyen, T.N.: Detecting semantic changes in Makefile build code. In: Software Maintenance (ICSM), 2012 28th IEEE International Conference on. pp. 150–159. IEEE (2012)
5. Almeida, B., Ananiadou, S., Bagnato, A., Barbero, A.B., Rocco, J.D., Ruscio, D.D., Kolovos, D.S., Korkontzelos, I., Hansen, S., Maló, P., Drivalos, N., Paige, R.F., Vinju, J.J.: OSSMETER: automated measurement and analysis of open source software. In: Procs. of the Projects Showcase, part of STAF 2015. pp. 36–43 (2015)
6. Androutsellis-Theotokis, S., Spinellis, D., Kechagia, M., Gousios, G.: Open source software: A survey from 10,000 feet. Foundations and Trends in Technology, Information and Operations Management 4(3–4), 187–347 (2011)
7. Cosentino, V., Izquierdo, J.L.C., Cabot, J.: Gitana: A sql-based git repository inspector. In: 34th Intl. Conf. on Conceptual Modeling (ER), pp. 329–343. Springer Intl. Publishing (2015)
8. van Deursen, A., Mesbah, A., Cornelissen, B., Zaidman, A., Pinzger, M., Guzzi, A.: Adinda: A knowledgeable, browser-based ide. In: Procs. of the 32Nd ACM/IEEE Intl. Conf. on Software Engineering - Volume 2. pp. 203–206. ICSE '10, ACM (2010)
9. Dyer, R., Rajan, H., Nguyen, H.A., Nguyen, T.N.: Mining billions of AST nodes to study actual and potential usage of Java language features. In: 36th Intl. Conf. on Software Engineering. pp. 779–790. ICSE'14 (June 2014)
10. Ebert, C., Gallardo, G., Hernantes, J., Serrano, N.: Devops. IEEE Software 33(3), 94–100 (2016)
11. Gousios, G.: The ghtorrent dataset and tool suite. In: Procs. of the 10th Working Conf. on Mining Software Repositories. pp. 233–236. MSR '13, IEEE Press (2013)

12. Hora, A., Valente, M.T.: apiwave: Keeping track of api popularity and migration. In: Procs. Intl. Conf. on Software Maintenance. pp. 321–323. IEEE (2015)
13. Jiang, Y., Adams, B.: Co-evolution of Infrastructure and Source Code: An Empirical Study. In: Procs. of the 12th Working Conf. on Mining Software Repositories. pp. 45–55. MSR '15, IEEE Press (2015)
14. Karlsson, E.A. (ed.): Software Reuse: A Holistic Approach. John Wiley & Sons, Inc. (1995)
15. Kolovos, D., Matragkas, N., Korkontzelos, I., Ananiadou, S., Paige, R.: Assessing the Use of Eclipse MDE Technologies in Open-Source Software Projects. In: Procs. of 2nd OSS4MDE at MODELS2015 (2015)
16. Landman, D., Serebrenik, A., Bouwers, E., Vinju, J.J.: Empirical analysis of the relationship between CC and SLOC in a large corpus of Java methods and C functions. Journal of Software: Evolution and Process (2016)
17. McIntosh, S., Adams, B., Hassan, A.E.: The evolution of Java build systems. Empirical Software Engineering 17(4-5), 578–608 (2012)
18. McIntosh, S., Nagappan, M., Adams, B., Mockus, A., Hassan, A.E.: A large-scale empirical study of the relationship between build technology and build maintenance. Empirical Software Engineering 20(6), 1587–1633 (2015)
19. Ponzanelli, L., Bacchelli, A., Lanza, M.: Seahawk: Stack overflow in the ide. In: Procs. of the 2013 Intl. Conf. on Software Engineering. pp. 1295–1298. ICSE '13, IEEE Press (2013)
20. Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., Lanza, M.: Mining stackoverflow to turn the ide into a self-confident programming prompter. In: Procs. of the 11th Working Conf. on Mining Software Repositories. pp. 102–111. MSR 2014, ACM (2014)
21. Ponzanelli, L., Bavota, G., Di Penta, M., Oliveto, R., Lanza, M.: Mining stackoverflow to turn the ide into a self-confident programming prompter. In: Procs. of the 11th Working Conf. on Mining Software Repositories. pp. 102–111. MSR 2014, ACM (2014)
22. Sharma, T., Fragkoulis, M., Spinellis, D.: Does Your Configuration Code Smell? In: Procs. of the Thirteenth Intl. Workshop on Mining Software Repositories. MSR'16, ACM (To appear)
23. Tonella, P., Potrich, A.: Reverse Engineering of Object Oriented Code (Monographs in Computer Science). Springer-Verlag New York, Inc. (2004)